CHARACTERIZATION OF THERMAL COUPLING IN CHIP MULTIPROCESSORS

A Thesis
Presented to
The Academic Faculty

By

Andrew Vanderheyden

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2014

CHARACTERIZATION OF THERMAL COUPLING IN CHIP MULTIPROCESSORS

Approved by:

Dr. Sudhakar Yalamanchili, Advisor
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Thomas Conte
Professor, School of Electrical and Computer Engineering and Computer Science
Georgia Institute of Technology

Dr. Saibal Mukhopadhyay
Associate Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 4, 2014

To my parents, my siblings, and my roommates, who have dealt with all of my complaining.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

For semiconductor processors temperature increases leakage current, which in turn increases the temperature of the processor. This increase in heat is seen by other parts of the processor since heat is diffusive across a processor die. In this way, cores are thermally coupled to one another such that when the temperature of one core increases, the temperatures of all cores on the same die can also increase. This increase in temperature and power consumption is not accompanied by any increase in performance. Cores on a chip can also be performance coupled to one another since cores can share data between them. These interactions between cores present new challenges to microarchitects who seek to optimize the energy consumption of a chip multiprocessor (CMP) comprised of multiple symmetric or asymmetric processing cores.

This thesis seeks to understand and model the impact of thermal coupling effects between adjacent cores in a chip multiprocessor starting with measurements with a commercial multi-core processor. The hypothesis is that the thermal coupling of compute cores will be influenced by the adjacent core's performance characteristics. Specifically, we expect thermal coupling is related to the nature of the workloads, e.g. compute intensive workloads will increase coupling over memory intensive workloads. However, we find that simpler parameters such as frequency of operation have more impact on coupling behaviors than the workload behaviors such as memory intensity or instruction retirement rates. A model is developed to capture thermal coupling effects and enable schemes to mitigate its impact.

CHAPTER 1

INTRODUCTION

With the end of Dennard scaling, the scaling of device feature size alone no longer guarantees sustaining the performance implications of Moore's Law. Increasing power densities have made it necessary to scale energy efficiency to sustain performance scaling. Thus, design regimes are shifting from performance oriented microarchitecture design to energy efficient and thermally efficient design. For a single core design, this problem is simply an issue of managing the processor core's voltage and frequency to optimize as the user needs for efficiency, performance, or temperature. The core's power and thermal characteristics are also dependent on its surroundings, but in the single core situation the core's surroundings are mainly cache and I/O controllers, which typically are relatively static in their power consumption compared to the processing core. Moving to multi- and many-core designs presents new challenges in managing heat and energy on a chip multiprocessor (CMP) due to transverse effects on adjacent cores

For semiconductor processors heat is diffusive across the die area. As a result, if one portion of the chip multi-processor is hotter than the rest of the chip, the other regions on the the chip will have their temperatures elevated. When this happens between regions on a chip, those two regions are said the be thermally coupled. Since different execution resources on a chip can share data or share other execution resources, different portions of the chip can also be performance coupled. In [2], the central processing unit (CPU) and graphics processing unit (GPU) are shown to be performance and thermally coupled. Using dynamic voltage and frequency scaling (DVFS) that is not aware of thermal coupling effects can exacerbate this effect, as also shown in [2]. The increasing heat diffusion causes increases in leakage power that reduce energy efficiency and degrade performance.

1.1 Performance vs. Power

For any processor the performance can be characterized as the product of the power consumed during operation and the processor's efficiency in operations per Joule, as in equation 1. To continue scaling performance within the same power budget, the processor efficiency must increase. Efficiency scaling is achieved in part via lithography scaling. However, as lithography scales down, power density (power dissipated per unit surface area) of a processor will increase due to the end of Dennard scaling. This makes the processor more challenging to cool.

$$Performance\ \left(\frac{ops}{sec}\right) = Power\ \left(\frac{Joules}{sec}\right) \times Efficiency\ \left(\frac{ops}{Joule}\right) \tag{1}$$

This increase in processor power density also has an effect on the interaction between cores on a chip multiprocessor. As it becomes more difficult to remove heat from a loaded processing core, more heat will dissipate through the neighboring regions of the chip. This is the effect that is known as thermal coupling between processing cores.

1.2 Thermal Coupling

As chip multiprocessors scale down in lithography, their thermal coupling interactions will become stronger. A simple analysis of a processor chip as a basic thermal circuit illustrates this phenomenon. Figure 1 shows the simplified thermal resistance circuit when examining only two cores on a chip multiprocessor. Q_{C1} represents the energy consumed by core 1 that is dissipated as heat. R_{1to2} represents the thermal resistance between core 1 and core 2. R_{HS} represents the resistance to the ambient via the heat sink. This value is the same for both cores, as the surface area in contact with the heat sink is the same for each core. All other chip surfaces are assumed to be adiabatic for the purposes of this analysis. The temperatures in the system are labeled as points in the schematic. T_{C1} is the temperature of core 1, T_{C2} is the temperature of core 2, and T_{amb} is the ambient temperature outside the

heat sink.

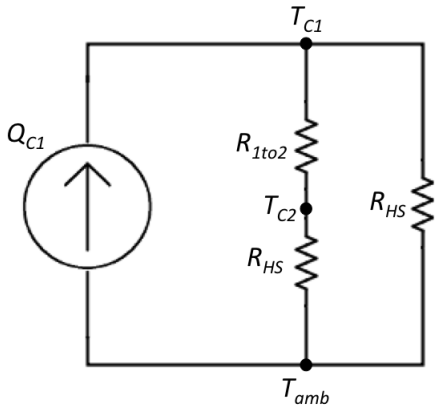

Figure 1: Basic thermal resistance circuit showing resistance between Core 1 and Core 2 (R_{C2}) and the resistance between the cores and the ambient via the heat sink (R_{HS})

To analyze how this circuit changes at smaller process nodes, an equation for each resistance based on physical properties must be found. For this simple model, R_{HS} is directly proportional to the surface area in contact with the core, as seen in equation 2.

$$R = \frac{x}{A \times k} \tag{2}$$

In this equation, x is the length of the material through which heat is diffusing, A is the cross-sectional area of the material, and k is the thermal conductivity of the material. Assuming the next-generation lithography is twice the areal transistor density of the current lithography node, observations can be made about how thermal resistance changes.

$$R_{C2} = \frac{x_{1to2}}{A_{1to2} \times k} \tag{3}$$

Equation 3 shows the the resistance between cores 1 and 2 as a function of the processor's layout. Assuming that the length and width of the processor die are equally affected

by the lithography scaling and that the thickness of the processor die does not change appreciably with the new lithography, the distance between cores 1 and 2 (x_{1to2}) and the cross sectional area between them (A_{1to2}) scale by a factor of $\frac{1}{\sqrt{2}}$. Examining equation 3, it can be shown that R_{C2} stays approximately constant as lithography scales.

Equation 4 shows the resistance between the individual cores and the heat sink. Under the assumption that die thickness does not change appreciably with lithography scaling, the distance between the heat source (transistors switching) and the heat sink (x_{HS}) stays constant. However, the cross sectional area between the heat source and the heat sink (A_{HS}) scales by a factor of $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} = \frac{1}{2}$. This causes the resistance between the core and the heat sink to increase by a factor of 2.

$$R_{HS} = \frac{x_{HS}}{A_{HS} \times k} \tag{4}$$

This is the driving force behind thermal coupling at next-generation lithographies: it becomes more difficult to remove heat via the heat sink, as the surface area of the equivalent processor layout decreases by a factor of 2. This results in more heat transferring to the neighboring regions of the chip multiprocessor.

1.3 Related Work

Work related to characterization of the thermal coupling effects on production chip multiprocessors is relatively sparse. In reviewing the literature, only a few papers were found that have the express purpose of characterizing the interactions of thermal coupling. In [3], the authors explore the coupling effects between a mobile application processor (AP), and the battery. In a mobile platform, the heat produced by the battery charging or discharging and the heat dissipated by the application processor will be coupled. Specifically, an increase in battery temperature due to the battery charging or discharging will cause an elevated temperature in the processor. However, the area of interest for this thesis focuses on thermal coupling within a single chip multiprocessor. In [4], simulations were run to

describe how the problem of thermal coupling becomes exacerbated at lower lithographies, but experiments on a real system were not performed.

Several papers have been written on how to mitigate the effects of thermal coupling. These papers largely present results from thermal coupling mitigation techniques such as DVFS state manipulation to balance the interactions between the CPU and GPU as in [2], floorplanning techniques to minimize the interactions between cores as in [5], and task migration schemes between cores such as those found in [6]. A feedback control framework specifically designed to account for thermal coupling between cores was presented in [7]. The results in [7] indicate that managing thermal coupling in an Intel Core 2 Duo system can provide advantages over existing DVFS control. The authors in [8] propose a new metric to assist in minimizing the effects of thermal coupling, which combines power, performance, and temperature.

1.4 Overview

Using a commodity multi-core processor, this thesis seeks to understand the thermal coupling between adjacent cores on a chip multiprocessor. The hypothesis of this thesis is that the thermal coupling of compute cores will be influenced by the adjacent core's performance characteristics. For example, when a core executes a compute-intensive workload, the neighboring core(s) will heat more than if a memory intensive workload is executed. The findings of this thesis are summarized in the list below.

- The temperature of a neighboring core is heavily dependent on the third order polynomial of the frequency of the loaded core.
- There is no discernible trend between workload characteristics and neighboring core temperature.
- The discrepancy in power consumption between the memory intensive benchmark and the compute intensive benchmark had no measurable effect on neighboring core

temperature.

- The work done by this processor does not vary appreciably with a memory-intensive versus a compute-intensive workload.

- Thermal coupling effects do not play a large role in the energy-delay product when only one core is loaded on this processor.

This thesis is organized into four main sections: characterization methodology, measurement, modeling, and conclusions. In characterization methodology the measurement interface used will be presented and discussed. In the measurement section the output of the measurement interface will be presented, and general trends will be discussed. In the modeling section the modeling infrastructure and results will be discussed. In the conclusions the implications of the results will be discussed, and a mitigation algorithm based on the results will be presented.

CHAPTER 2

CHARACTERIZATION METHODOLOGY

This section will present the infrastructure and methodology used to characterize thermal coupling effects on a production chip multiprocessor. It begins by presenting the processor used to collect the measurement data, followed by the measurement infrastructure used to collect the data. Then the benchmarks used to generate the measured data are discussed. Finally, this section presents the modeling infrastructure used to create a model from the data and concludes with a concise overview of the steps included in the thesis.

2.1 Processor

To generate results for a modern commodity microprocessor, measurements for power and temperature were taken on an Intel® Core™ i5-3470. Figure 2 shows the layout of the Intel® Ivy Bridge microarchitecture.

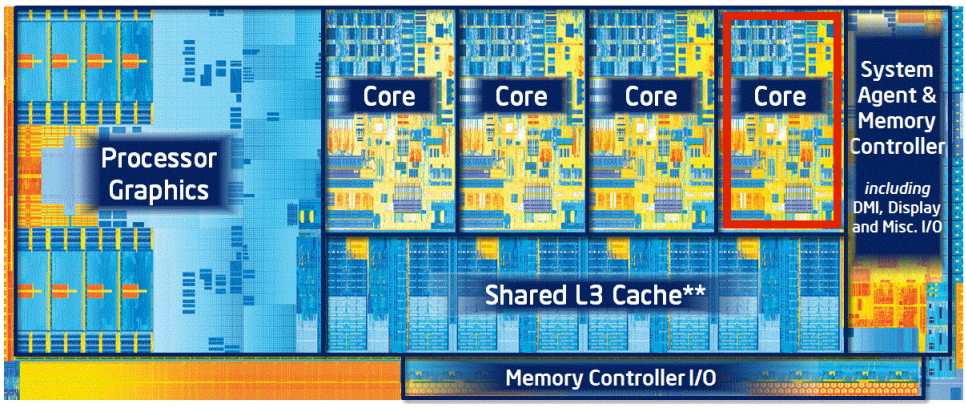

Figure 2: Labeled layout of Intel® Ivy Bridge CPU [1]

The memory controllers lie along the bottom of the processor just below the L3 Cache and along the right hand side of the die. The cores lie in the middle of the die as shown; however, data on how cores are mapped to the system is not available. In the course of the measurements, it was determined that Cores 2 and 3 are neighbors and that Core 0 is substantially farther from Core 3 than Core 2. The focus of this thesis is characterizing the

interactions between two neighboring cores, so the interaction between Cores 2 and 3 was examined. This was done by forcing the benchmark in question to only execute on Core 3, and then examining how Core 2's temperature responded to Core 3's processing load. The core outlined in red on the figure is assumed to be Core 3 for this analysis, with Core 2 assumed to be immediately to its left.

2.2 Measurement Infrastructure

Power measurements were gathered using the Running Average Power Limit (RAPL) registers present in the microprocessor, as documented in [9]. These power sensors are refreshed every 1 ms and have a resolution of 15.2 nJ [10]. The RAPL registers provide power data on a per-plane basis. Specifically, the Ivy Bridge microarchitecture is broken into two planes: power plane 0 (PP0) and power plane 1 (PP1). PP0 contains all cores and caches, while PP1 contains the rest of the chip's resources ("uncore", memory controller(s), graphics processing unit, etc.). Since these registers are only 32 bits long, they are subject to overflow, which is not taken into account by the counters. However, no overflow errors were seen using these registers. The energy values reported by RAPL are counters that must be modified by the energy units reported by RAPL, which was always the same for these measurements (15.2 nJ).

Performance and temperature measurements were taken using Intel® Performance Counter Monitor (PCM). PCM allows for measurement of various parameters of the processor as a whole and of the individual cores via an API. It also allows for measuring power consumed by the processor at the socket level. Direct access to the RAPL registers is needed to obtain power statistics broken down into power planes.

Both RAPL and Intel's PCM utility rely on model-specific registers (MSR's) to read their counters. These MSR's are registers present on the chip that allow the operating system and programs to get online performance data. The reported temperature values for each core are the offset from $T_{j_{max}}$, which represents the maximum allowed temperature for

the chip [9]. For Ivy Bridge, $T_{j_{max}}$ is 105 °C. The temperature sensors have a resolution of 1 °C. Equation 5 shows the relationship between actual temperature and the reported temperature value from the model specific register (T_{offset}). Experimental results indicate the temperature sensors are refreshed approximately every 1 ms.

$$T = T_{j_{max}} - T_{offset} \tag{5}$$

For gathering both fine-grained power via RAPL and performance metrics via PCM, a short program was written. The total list of metrics gathered by the program, along with their descriptions, is presented in Table 1. This program samples the counters every 10 ms and reports the values for each metric.

Table 1: Metrics captured via measurement interface

Metric	**Definition**
READ	Total data read from memory (Gigabytes)
WRITE	Total data written to memory (Gigabytes)
INST	Total instructions retired
ACYC	Total number of unhalted core clock cycles
Mticks	Total number of invariant time ticks (Mega-ticks)
ProcEnergy	Total Joules consumed at the socket level via PCM
TEMPc0	Temperature of Core 0
TEMPc1	Temperature of Core 1
TEMPc2	Temperature of Core 2
EXECc3	Instructions per nominal CPU cycle of Core 3
IPCc3	Instructions per cycle of Core 3
FREQc3	Relative frequency of Core 3
AFREQc3	Relative frequency of Core 3 while in active state
L3MISSc3	Total number of L3 misses for Core 3
L2MISSc3	Total number of L2 misses for Core 3
L3HITc3	L3 hit rate for Core 3
L2HITc3	L2 hit rate for Core 3
L3CLKc3	Total number of clock cycles lost due to L3 misses
L2CLKc3	Total number of clock cycles lost due to L2 misses
TEMPc3	Temperature of Core 3
EU	Units of consumption (Joules)
PP0	Energy consumed by Power Plane 0 via RAPL
PP1	Energy consumed by Power Plane 1 vie RAPL

Relative frequency in the description above is the frequency at which the core operates relative to the nominal frequency. This value is a scalar between 0.5 and 1.1875 for a nominal frequency defined for the Intel® Core™ i5-3470 of 3.2 GHz. The measurements *FREQc3* and *AFREQc3* are calculated by comparing the total cycles counted from the last sample for the core's clock to the total cycles counted by the invariant time stamp counter (invariant TSC), as defined in [9]. This invariant TSC is also used to count number of ticks since the last sample for the *Mticks* measurement, and as the number of cycles in the computation of *EXECc3*.

2.3 Benchmark Infrastructure

Two benchmarks were chosen from the PARSEC benchmark suite to be extensively examined for this research. The intent in choosing two benchmarks was to represent a benchmark set that would be considered more memory intensive and less compute intensive, and one that would be considered more compute intensive and less memory intensive. To that end, the benchmarks *canneal* and *dedup* were chosen for the compute-intensive and memory-intensive applications, respectively, as shown in [11] and [12]. These benchmarks were also chosen because their "*Native*" benchmarks provide a long execution time, which is beneficial for gathering both transient and steady state effects in a single benchmark run.

The benchmarks were run on a system using commodity computer parts, with Ubuntu 12.04 as the operating system. A heat sink was attached to the processor, specifically the Cooler Master Hyper 212 EVO. There was no fan attached to the heat sink, but other fans in the system were present. To set the frequency, the multiplier for the CPU frequency as a function of the base clock was set from 16 to 38 in increments of 2, giving frequencies from 1.6 GHz to 3.8 GHz in 200 MHz increments. Turbo Boost and all other various power saving utilities that could be accessed were disabled. The CPU core voltage was set at 1.1 V for all frequencies.

2.4 Modeling Infrastructure

After gathering the performance measurements, the modeling of the data was performed using Statistical Analysis Software (SAS) version 9.4. The temperature data was first smoothed with a basic triangular moving average using a window size of 3. The intent of smoothing the temperature data within each benchmark run was to offset the relatively low resolution of 1 °C of the on-die temperature sensors.

Since the measurements were gathered at fixed time intervals, the lowest frequencies had over twice as many observations as the highest frequency. As a result, a subset of observations from each benchmark - excluding the highest frequency - was randomly selected without replacement from each run of the benchmark. The goal of this re-sampling was to include the same number of observations from each frequency to prevent skewing the model building steps. This sampling was done after computing both the moving average of the loaded core's temperature and the moving sum of instructions retired (*cINST*). The number of observations in the highest frequency benchmark run was used as the number of observations to randomly select from the other benchmark runs. This was done independently for *canneal* and *dedup*, as the execution time of each varies greatly.

To create a model for the data, only performance metrics were considered. Energy and temperature measurements were omitted from the model construction. After removing these measurements from the data set, feature selection was used to determine which metrics were most closely tied to the neighboring core's temperature. To determine if higher order derived metrics should be included in the feature set, the data was visually inspected. This was done by plotting each performance metric against neighboring core temperature and inspecting the data for general trends. If the data presented a "saddle" or inflection point, the cubed and squared values of that metric was included in the set. Equation 6 shows the basic third order polynomial function that is centered at *x*-*y* coordinates (*a*,*b*).

$$y - b = (x - a)^3 \tag{6}$$

Equation 7 shows the result if equation 6 is expanded, and the y variable is isolated. This will result in a term that is a function of x^2, and a term that is a function of x for all values of $a \neq 0$. Due to this interaction, when the cubed value of a metric was included in a regression model, the squared and linear values were also included.

$$y = x^3 - 2ax^2 + 2a^2x - a^3 + b \tag{7}$$

Factor analysis was performed on the resulting data. This involves principle components analysis, and a subsequent rotation of a subset of the principle components. The rotation is done in an attempt to force the principle components to load on a few metrics where most of the data's variance lies.

Distinct from factor analysis, stepwise feature selection was performed. Stepwise feature selection is similar to forward feature selection in that it begins by selecting the feature with the largest F statistic, which reflects the metric's contribution to the model if it is included. It differs from pure forward selection by adding the ability to remove features already in the selected subset if that feature's F statistic drops below a threshold as a result of the addition of a new feature.

After examining the results from both feature selection techniques, a selection of models was created for each benchmark individually using the neighboring core's (Core 2) temperature as the dependent variable. The benchmarks were left separate in an effort to determine the differences in how the neighboring core's temperature responds to a varying processing load. All regressions run were linear multiple regressions, as the desired non-linearities were included in the metric generation step.

To test the models that were constructed in the regressions, a holdout analysis was performed. For each model, a randomly selected proportion of the data set that is approximately 10% of the data was held out of the model construction for this step. This model based on part of the data set was then used to predict the values of the holdout data set, and the R^2 value for the holdout and the model were compared to determine whether they were

close to each other.

Additionally, the final models constructed via multiple regression were tested by predicting values of a third benchmark (*x264*). The observations from the *x264* benchmark were sampled the same way the observations from *dedup* and *canneal* such that there were an equal number of observations from each of the twelve frequencies tested. This was done to test the validity of broader application of these models. These predicted values were then analyzed to find their coefficient of determination (R^2). This coefficient was then compared to the coefficient of determination for the two finalized models to determine whether if either of the two finalized models could accurately model *x264*.

2.5 Overview

Figure 3 shows the overall flow diagram for this characterization methodology. The analysis begins with collecting data from the benchmark runs using 12 different frequencies for the processor. Then, the results are examined for general trends in the data to determine which (if any) higher order derived features should be included in the analysis. Feature selection and factor analysis are performed in order to inform the modeling building and investigation process. Finally, a final ”best” model is selected.

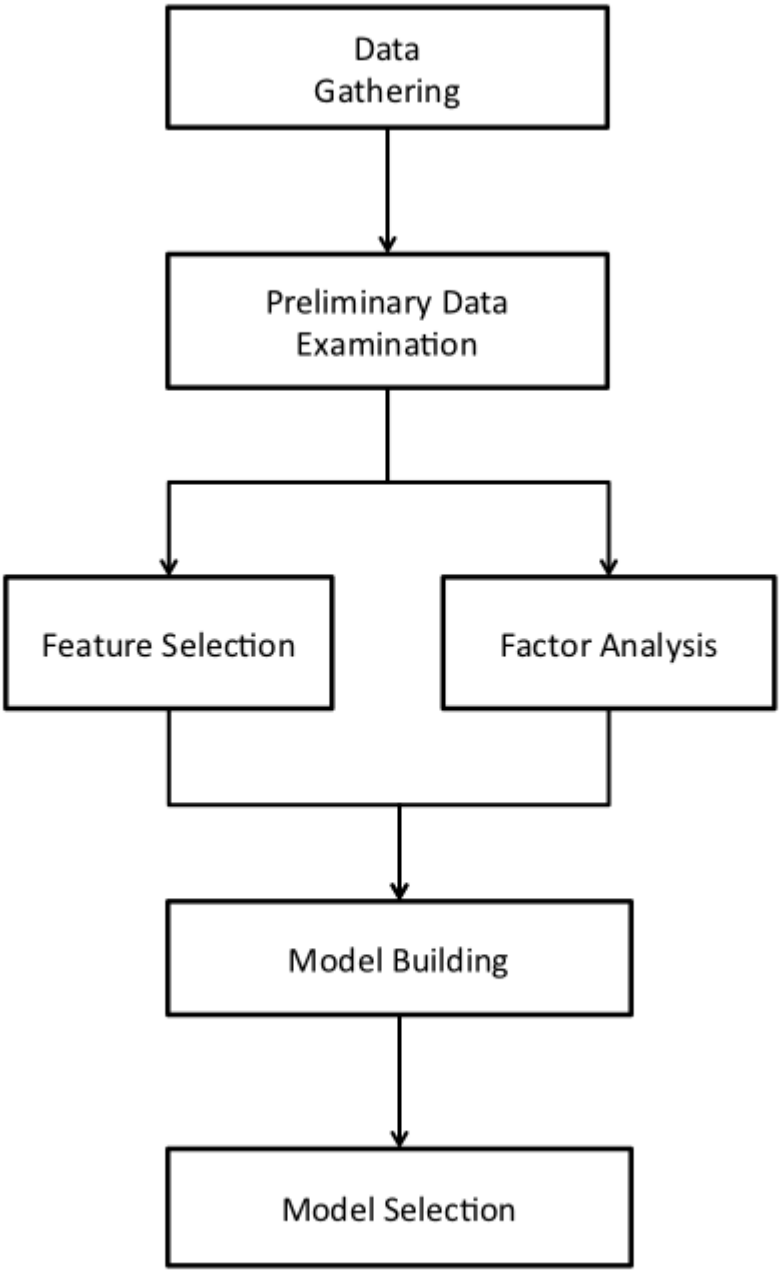

Figure 3: Flow diagram for characterization methodology.

CHAPTER 3

MEASUREMENT

This chapter contains results for both temperature and power for both benchmarks generated using the measurement infrastructure described in Chapter 3. General trends in the data are discussed.

3.1 Temperature Measurements

An initial examination of the temperatures of other cores on the chip while a processing load was applied to one core was performed. Figure 4 shows the results from the *canneal* benchmark. Figure 5 shows the temperature data for each frequency from the *dedup* benchmark. The larger circles indicate the average temperature for that frequency.

The temperature clearly trends upward as frequency scales, but it doesn't appear linear. The trend appears to be a polynomial, exponential, or possibly a logarithmic function. Additionally, it is clear that Core 1 has some error in it. This was true for all benchmark runs that were examined, as well as at idle: Core 1 reports slightly higher temperatures than the rest of the cores. At higher frequencies, it doesn't heat as much due to Core 3's load, which may indicate that the particular reading from Core 1 may be incorrect by a static offset. It is also possible that Core 1 is situated next to the graphics processing unit or next to the memory controller (the left and right ends of the core array seen in figure 2). Since the only results being examined are the results of Core 2 and Core 3, this error is ignored.

3.2 Power Measurements

Figure 6 shows the average power consumed for the *canneal* and *dedup* benchmarks. For both benchmarks, the relationship between average power and frequency appears approximately linear. *Dedup* appears to consume slightly more power on average than *canneal* in

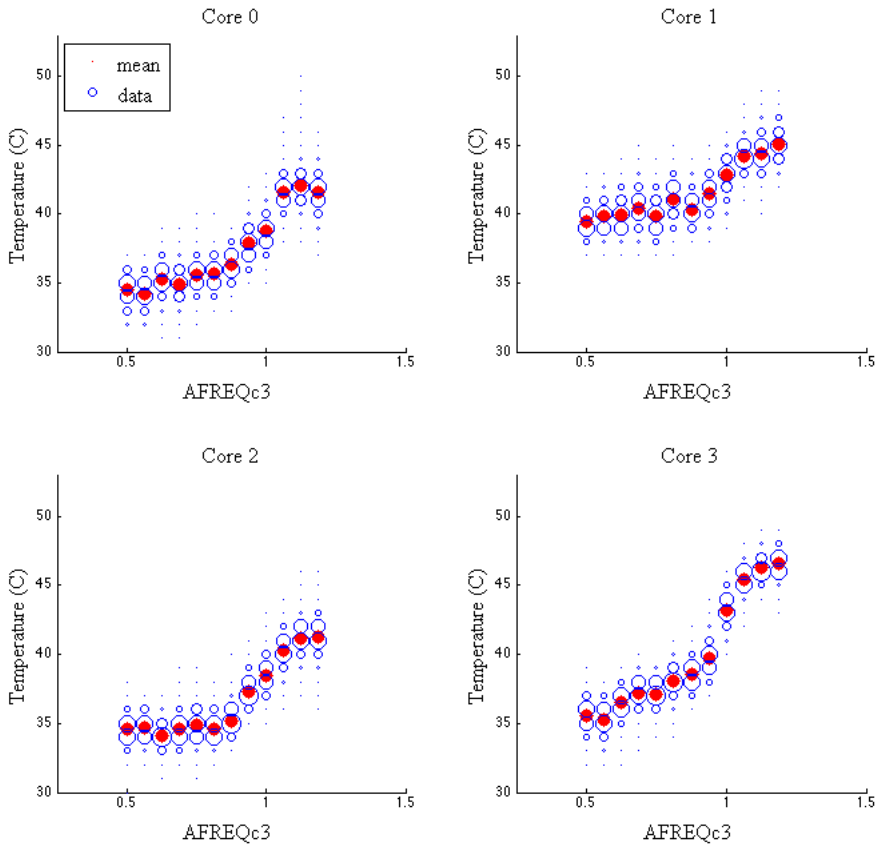

Figure 4: Temperature vs. frequency for the *canneal* benchmark.

power plane 0 (PP0). PP0 contains all processors and caches. Since *dedup* has more memory accesses than *canneal*, it is possible that the additional power consumed is due to more frequently accessing the L3 cache and memory controllers, which neighbor each other as seen in figure 2. More frequently using those two elements could raise the temperature of the L3 cache, which would lead to higher leakage power for the L3 cache.

Figure 7 shows the total energy consumed by the *canneal* benchmark, and figure 8 shows the same results for the *dedup* benchmark. It is readily seen that *dedup* has much lower total energy consumed than *canneal* due to the benchmark's lower execution time, but the total energy consumed during the *dedup* benchmark is actually decreasing from 1.6 GHz to around 3.4 GHz. In contrast *canneal*'s total energy increases with frequency. This

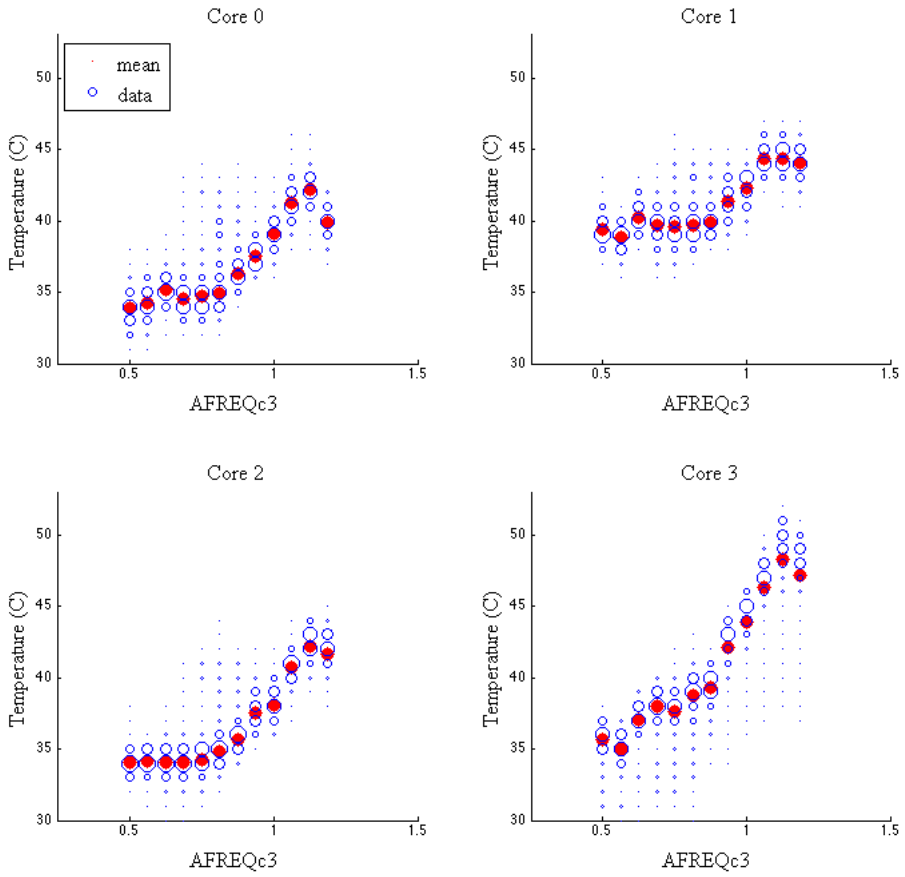

Figure 5: Temperature vs. frequency for the *dedup* benchmark.

is likely a result of the fundamental difference in their workloads (*canneal* being compute-intensive and *dedup* being memory-intensive).

Figure 9 shows the energy-delay product for the *canneal* benchmark, as described by equation 8, where E is the total energy consumed by the processor during a benchmark run, and T is the total time taken to complete the benchmark, and P_{ED} is the resulting energy-delay product. The energy-delay product for *canneal* seems to have one general trend: the plot appears to be quasi-parabolic, with a minimum around 3.0 GHz. This result indicates that the optimal frequency for the *canneal* benchmark can be found. In equation8, E is the energy consumed by PP0, and T is the total time required to complete the benchmark.

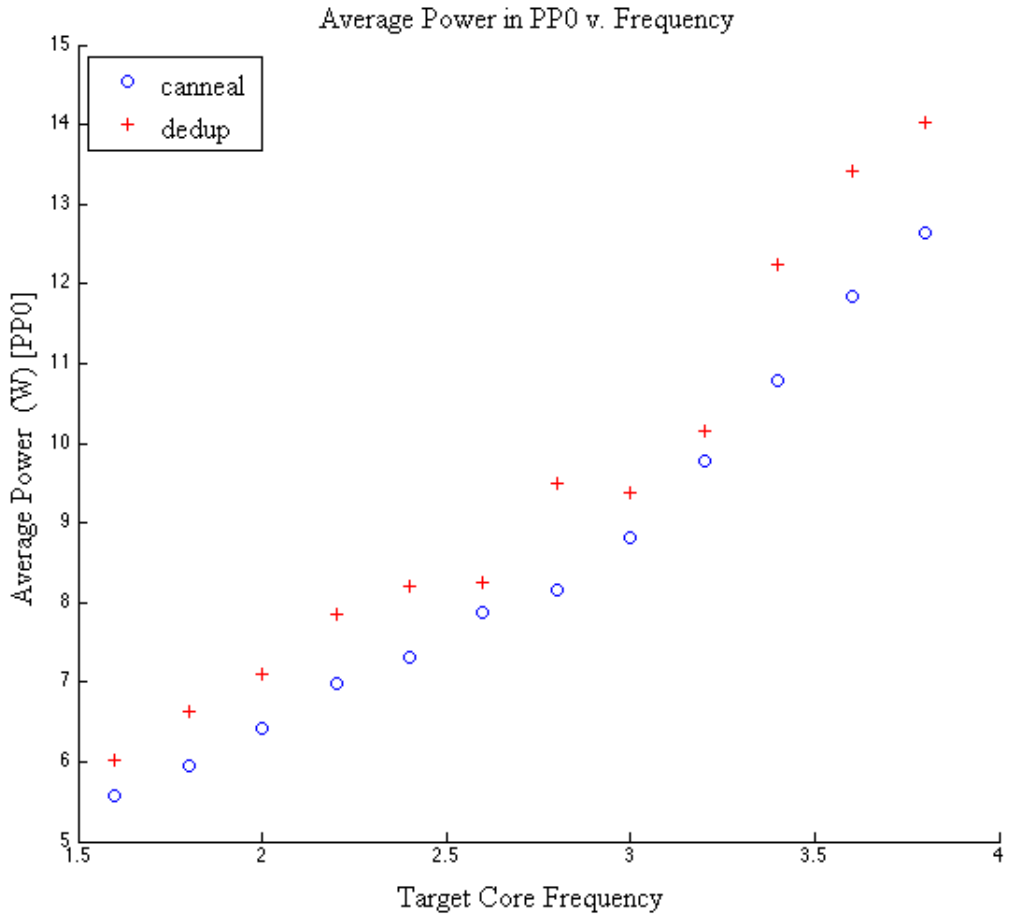

Figure 6: Average power vs. frequency for the *canneal* and *dedup* benchmarks.

$$P_{ED} = E \times T \tag{8}$$

Unlike the energy-delay product for the *canneal* benchmark, the product for the *dedup* benchmark appears to decrease as frequency increases, with no visible minimum for this frequency range, as shown in figure 10. This suggests that the *dedup* workload can be executed at the maximum supported frequency by this processor to yield a minimized energy-delay product for PP0.

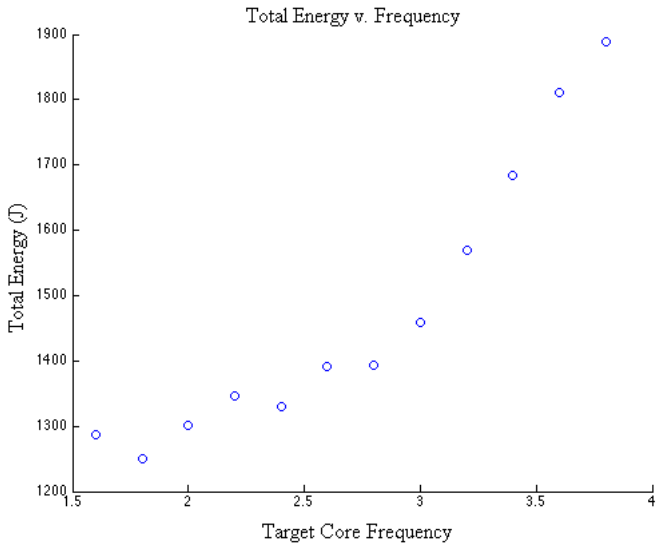

Figure 7: Total energy consumed vs. frequency for the *canneal* benchmark.

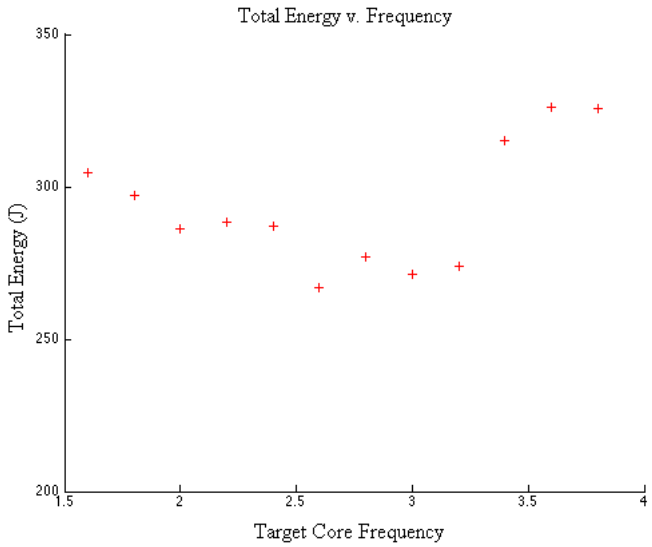

Figure 8: Total energy consumed vs. frequency for the *dedup* benchmark.

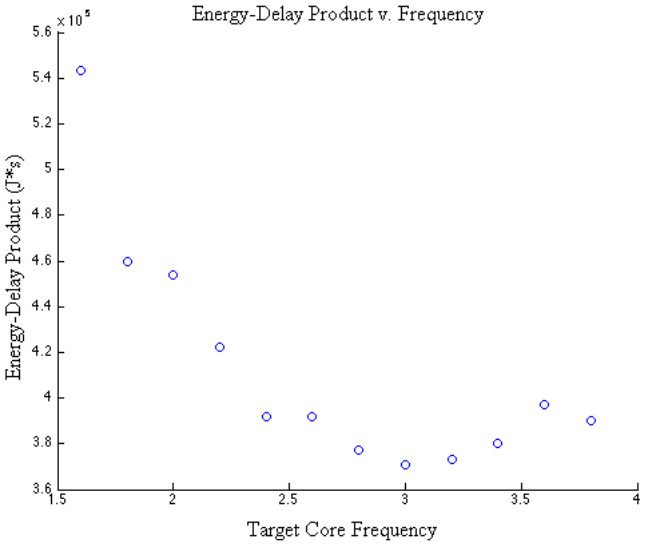

Figure 9: Energy-delay product vs. frequency for the *canneal* benchmark.

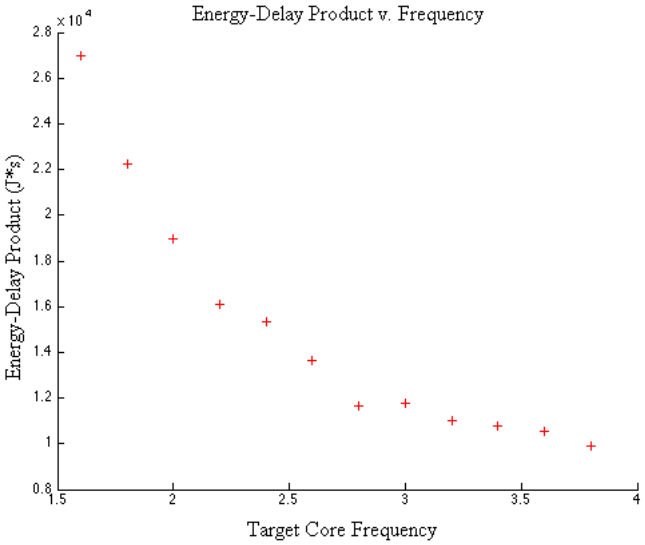

Figure 10: Energy-delay product vs. frequency for the *dedup* benchmark.

CHAPTER 4

MODELING

This chapter contains the modeling results for these two benchmarks. It begins by discussing the two feature selection methods used to inform the model building process. It then presents a small subset of the models tested for this thesis, and compares and contrasts them. Finally, it presents the final chosen model for these two benchmarks and discusses the fit of that model in more detail.

4.1 Feature Selection

Two methods of feature selection were explored as a means of informing the choice of which metrics to include in the final model for each of the benchmarks. The results for factor analysis can be found in Appendix A, and the results for sequential feature selection can be found in Appendix B.

Initial examination of the data indicated a possible cubic behavior in the *AFREQc3* and *FREQc3* variables, as well as possible quadratic trends in both *IPCc3* and *INST*. As a result, the cubed and squared values of *AFREQc3* and *FREQc3* were included in the analysis, as well as the squared values of *IPCc3* and *INST*. The cubed values are named "****cub*" and the squared values are named "****sqr*," where the "****" represents the original variable name.

4.1.1 Factor Analysis

The results of the factor analysis from the *canneal* benchmark can be found in Appendix A. The tables in the appendix contain the factor loading for the first six principle components of the separate benchmarks' data sets. These components were found using principle components analysis, and the resulting vectors were rotated to maximize their loading on as few variables as possible.

Unfortunately, no significant loading on any individual metric was found for either the

canneal or *dedup* benchmark set. This indicates that the variance in the data does not lie mainly in one component, and suggests that the metrics that load together may or may not be linearly independent. As a result, factor analysis was not used to inform the selection of the models for the subsequent steps.

4.1.2 Sequential Feature Selection

The steps for sequential feature selection using the stepwise method can be found in Appendix B. The results for both *canneal* and *dedup* show that the cubed frequency values are the first metrics selected, followed in both cases by a regular (non-squared or cubed) frequency metric. Using these two metrics, the model R^2 approaches 0.9 for both *canneal* and *dedup*. Selecting the cubed frequency first, combined with the cubed frequency having such a large incremental R^2 indicates that the neighboring core's temperature is heavily dependent on the operating frequency of the loaded core.

After selecting the first two metrics, the two selection steps diverge. Feature selection for the *canneal* benchmark selects *FREQc3sqr* next, and then selects two different measures of instruction throughput: *cINST* and *EXECc3*. This is likely an artifact of the *canneal* benchmark having relatively few memory instructions, which leads to a smaller variation along the memory and cache related metrics.

The third selection step for *dedup* selects *L3CLKc3*, which is a cache-based metric in that it counts the total number of clock cycles lost to misses in the L3 cache. After selecting this, the feature selection algorithm selects *FREQc3cub*, which further indicates a strong trend in this data with the cubed value of the relative frequency. After selecting the second cubed value of relative frequency, the algorithm selects another cache-based metric: *L2HITc3*.

In both the *canneal* and *dedup* benchmarks, adding more features after the initial three or four metrics does not yield an appreciably increase in model performance. This shows that the trends present in this data can be accurately modeled using relatively few performance metrics.

4.2 Model Building

For each benchmark three models are presented. The first contains simply the first few metrics from the feature selection step. The second contains a benchmark specific subset of performance variables, excluding the frequency variables. This was explored to determine whether the temperature of the neighboring core could be explained well via metrics outside of frequency. The final model presented shows the final model selected for each benchmark.

4.2.1 *canneal*

For the *canneal* benchmark using *AFREQc3* compared to *FREQc3* made no difference in the model. The first model is simply the first 4 metrics from the feature selection step. It provides a good R^2 value, but is a function of 3 different measurements (*FREQc3*, *AFREQc3*, and *EXECc3*). The second model shows that without frequency, the model fit is comparatively poor. As a result, a model that includes frequency was chosen for the final model. In this case, good results were obtained by only using a third order polynomial of *AFREQc3*.

4.2.2 *dedup*

The regression models for the *dedup* data are shown in table 3. The first model in this table shows the fit metrics and coefficients for the variables for the model using the first four metrics selected via the stepwise feature selection method. The fit is good, but the model is not optimal because it is a function of four different metrics. The second model shows the quality of model that can be obtained using non-frequency variables. The fit for this model (R^2 of 0.69) is fairly good, but not nearly as well-fitted as a model that contains a frequency variable. The final model in the table contains only the components needed to create a third order polynomial on the *AFREQc3* variable. This model has very good fit, and was chosen as the "best" model for this benchmark.

Table 2: Regression models for *canneal*

Model 1	**Model Performance**					
	Root MSE	0.80722	R-Square	0.9142		
	Dependent Mean	68.24842	Adj R-Sq	0.9142		
	Parameter Estimates					
	Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >| *t* |
	Intercept	1	41.91042	0.13683	306.3	<0.0001
	FREQc3cub	1	38.36929	0.25085	152.96	<0.0001
	AFREQc3	1	106.79309	0.52087	205.03	<0.0001
	FREQc3sqr	1	-120.42407	0.63665	-189.15	<0.0001
Model 2	**Model Performance**					
	Root MSE	1.86866	R-Square	0.5401		
	Dependent Mean	68.24842	Adj R-Sq	0.5401		
	Parameter Estimates					
	Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >| *t* |
	Intercept	1	71.27702	0.03545	2010.55	<0.0001
	EXECc3	1	-28.24045	0.12166	-232.12	<0.0001
	cINST	1	0.01463	0.00055835	26.21	<0.0001
	READ	1	-84.49979	2.36534	-35.72	<0.0001
	L2HITc3	1	-5.36898	0.12715	-42.23	<0.0001
	WRITE	1	6.71957	3.08209	2.18	<0.0001
	IPCc3sqr	1	63.27674	0.28754	220.06	<0.0001
	IPCc3cub	1	-40.32039	0.2471	-163.18	<0.0001
Model 3	**Model Performance**					
	Root MSE	0.80723	R-Square	0.9142		
	Dependent Mean	68.24842	Adj R-Sq	0.9142		
	Parameter Estimates					
	Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >| *t* |
	Intercept	1	41.91047	0.13683	306.3	<0.0001
	AFREQc3cub	1	38.36914	0.25085	152.96	<0.0001
	AFREQc3	1	106.79284	0.52087	205.03	<0.0001
	AFREQc3sqr	1	-120.42373	0.63665	-189.15	<0.0001

Table 3: Regression models for *dedup*

Model 1	**Model Performance**					
	Root MSE	1.05985	R-Square	0.8872		
	Dependent Mean	68.229	Adj R-Sq	0.8872		
	Parameter Estimates					
	Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >| *t* |
	Intercept	1	71.20358	0.02713	2624.86	<0.0001
	AFREQc3cub	1	-7.56564	0.02583	-292.86	<0.0001
	FREQc3	1	2.12146	0.05531	38.36	<0.0001
	L3CLKc3	1	0.63652	0.01428	44.58	<0.0001
	cINSTsqr	1	0.00001052	5.5911e-7	18.82	<0.0001
Model 2	**Model Performance**					
	Root MSE	1.7524	R-Square	0.6916		
	Dependent Mean	68.229	Adj R-Sq	0.6916		
	Parameter Estimates					
	Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >| *t* |
	Intercept	1	63.62116	0.12732	499.68	<0.0001
	L3CLKc3	1	0.22298	0.03486	6.4	<0.0001
	cINSTsqr	1	-0.00017488	2.020e-6	-86.62	<0.0001
	L2HITc3	1	-2.69275	0.10816	-24.9	<0.0001
	IPCc3	1	7.08791	0.24474	28.96	<0.0001
	IPCc3sqr	1	-0.67276	0.07602	-8.85	<0.0001
	cINST	1	0.02841	8.4939e-4	33.45	<0.0001
Model 3	**Model Performance**					
	Root MSE	1.04841	R-Square	0.8896		
	Dependent Mean	68.229	Adj R-Sq	0.8896		
	Parameter Estimates					
	Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >| *t* |
	Intercept	1	44.1245	0.45858	96.22	<0.0001
	AFREQc3	1	103.24458	1.75127	58.95	<0.0001
	AFREQc3sqr	1	-119.82338	2.14831	-55.78	<0.0001
	AFREQc3cub	1	38.52669	0.8494	45.36	<0.0001

4.3 Model Fit

The results of the model building step indicate that for both compute-intensive and memory-intensive benchmarks, frequency is the driving force behind thermal coupling effects. Furthermore, they indicate that *AFREQc3* is effectively interchangeable with *FREQc3* for the *canneal* benchmark, but present slightly different models for *dedup*. As per Intel's documentation, these metrics are computed as they are show in equations 9 and 10.

$$FREQ = \frac{\text{unhalted clock ticks}}{\text{invariant timer ticks (includes Intel Turbo Boost)}} \tag{9}$$

$$AFREQ = \frac{\text{unhalted clock ticks}}{\text{invariant timer ticks while in C0-state (includes Intel Turbo Boost)}} \tag{10}$$

For this study Intel's Turbo Boost was disabled. The values of *AFREQc3* and *FREQc3* are very similar for the *canneal* benchmark since this benchmark would typically spend relatively little time halted and waiting for a response from memory, as it's more compute-intensive than memory-intensive. Conversely, for the *dedup* benchmark, a slightly worse model was created using a third order polynomial of *FREQc3* than when using a third order polynomial of *AFREQc3*. This is likely because *FREQc3* would typically not be the same as *AFREQc3* for a benchmark that is more memory-intensive, rather than compute-intensive, and it spends more time waiting on data from memory and in a halted state.

Based on the results from the feature selection step, the model for both *canneal* and *dedup* that was selected as the optimal model was the third order polynomial of *AFREQc3*. This model minimizes the number of independent variables (there is only one), and provides a good enough fit that adding more features to the model is unnecessary.

4.3.1 *canneal*

The models that were fit to the *canneal* data turn out to perform extremely well. An R^2 of approximately 0.9 or larger for measured data is a very good fit for the model. Figure 11 displays the fitted model and the data gathered. The line represents the fitted data, and the circles are the measured data. A bi-variate histogram of the data was created and used to vary the marker size for each blue marker in order to show how the data clusters. The different size markers represent the number of observations that were found in that bin.

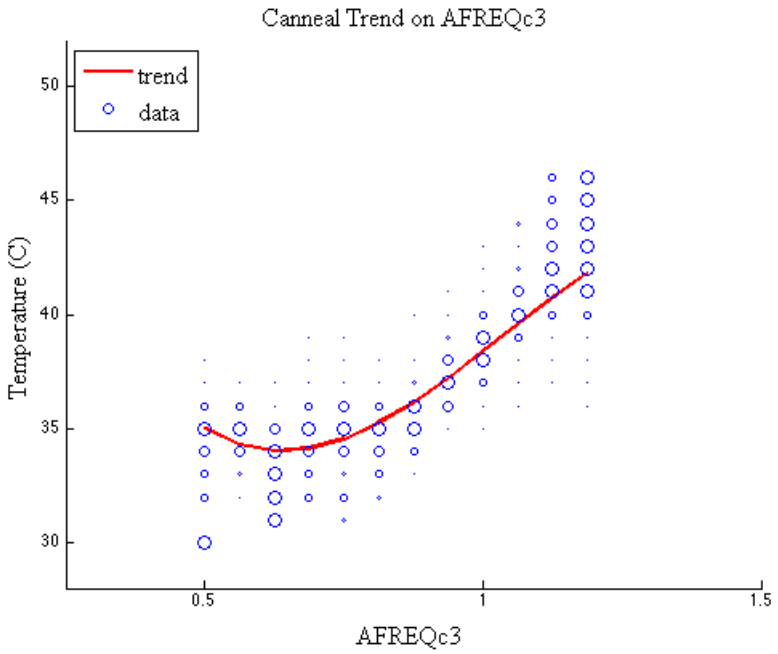

Figure 11: Neighboring core temperature vs. *AFREQc3* for *canneal*

This figure clearly shows the 3rd order behavior of the resulting fitted model. At low frequencies, it's likely the small increase in temperature is due to some measurement error, as it is within the 1 °C resolution of the digital temperature sensor. Figure 12 shows the predicted value and 95% confidence intervals for the data, along with the residual values. This plot is *wTEMPc2* vs. *AFREQc3*. Note that *wTEMPc2* is the weighted average value of *TEMPc2*, and *TEMPc2* is defined as the offset from $T_{j_{max}}$. The actual temperature of the chip is defined in equation 5, where T_{offset} represents the value of *wTEMPc2*.

Notably, the residuals for this model appear consistent for the center portion of the

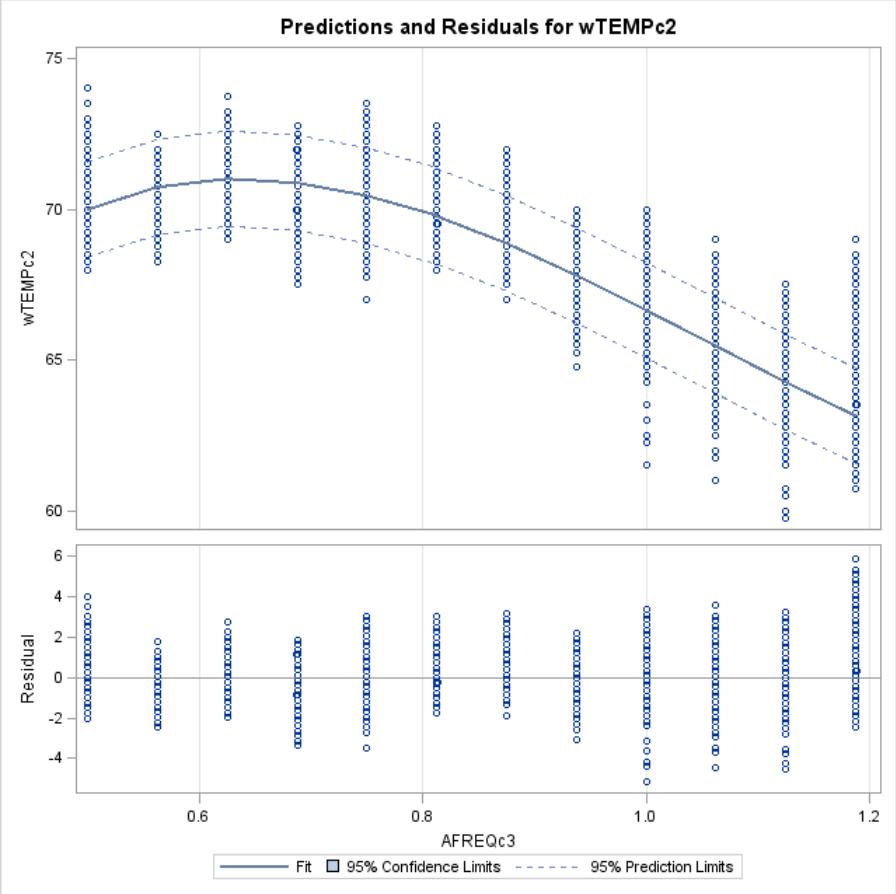

Figure 12: Fit statistics for 3rd order polynomial model of *AFREQc3* for *canneal*

model, but deviate slightly for the highest and lowest frequencies. This observation indicates that this model is a good fit for the data. Specifically, there do not appear to be any systemic issues in this model that could be solved by increasing the model complexity.

Holdout analysis was performed on these results by fitting the data to 90% of the observations and predicting the remaining 10%. For the *canneal* benchmark, the holdout analysis yielded a coefficient of determination for the held out data of 0.9116. This result confirms that the model performs as well as the finalized model's R^2 suggests.

4.3.2 *dedup*

Figure 13 shows the fitted data for the *x264* benchmark plotted against the measured data from that benchmark. The data shown as circles are the measured data, and a bi-variate histogram of the data was used to inform the marker sizes to show the data clustering. The

line is the trend line plotted by the fitted model.

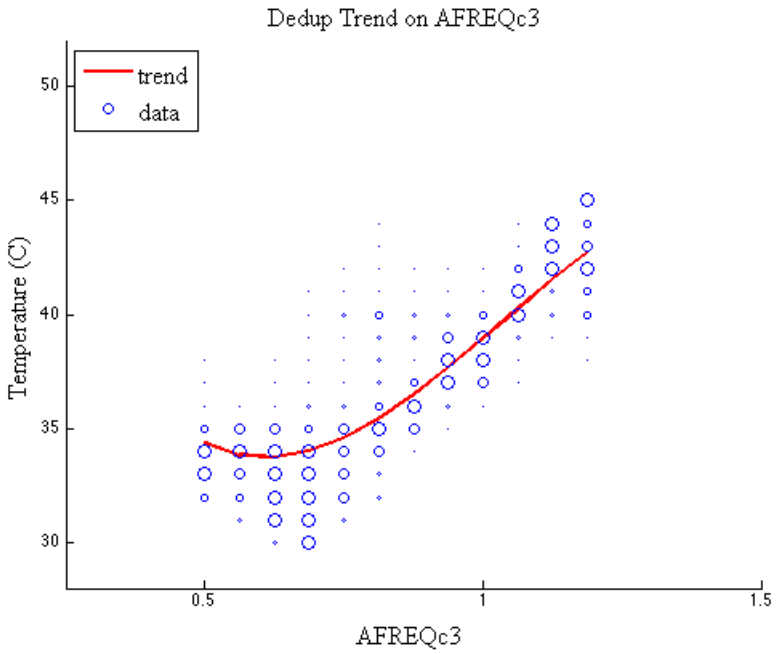

Figure 13: Neighboring core temperature vs. *AFREQc3* for *dedup*

This plot looks very similar to the plot generated for *canneal*. The trend appears fairly similar, and it appears to fit the data well. The R^2 for model 3 from table 3 of 0.8896 reflects this good fit. The model is not as well-fitted as the model that was found for *canneal*, but the difference in the R^2 statistic is small enough that it can be disregarded.

The plots of the residuals for this model are in figure 14. Another contributor to the decreased R^2 for this model when compared to the model for *canneal* is the fact that the loaded processor core does not stay at its highest available frequency while it is active. It is likely that this is due to the benchmark being forced to wait for small amounts of time in the active state for results from memory, but not waiting so long that the core is able to switch to a halted state. This is not the case for compute-intensive workloads, as they do not have to wait on results from memory as often. However, *dedup*'s *FREQc3* and *AFREQc3* models still performed very similarly.

Holdout analysis was performed on these results by fitting the data to 90% of the observations and predicting the remaining 10%. For the *dedup* benchmark, the holdout analysis

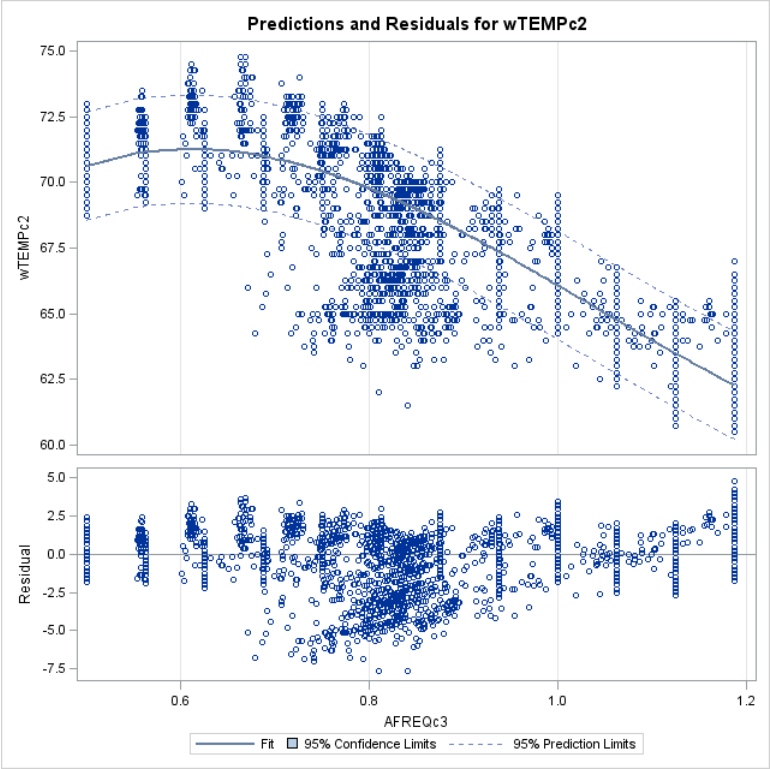

Figure 14: Fit statistics for 3rd order polynomial model of *AFREQc3* for *dedup*

yielded a coefficient of determination for the held out data of 0.8827. This result confirms that the model performs as well as the finalized model's R^2 suggests.

4.3.3 Model Comparison

The plot showing both trend lines is presented in figure 15. The blue line is the fitted trend for the *canneal* benchmark, and the red line is the fitted trend for the *dedup* benchmark.

These trends are very close to one another. So close that they are within the resolution error of the digital temperature sensor, which measure at a resolution of 1 °C. As a result, they can be said the be more or less the same trend, which has implications on this thesis's hypothesis. The primary implication is that the temperature of the neighboring core trends heavily with the frequency of the loaded.

Also importantly, this model and data indicate that the lowest frequency (1.6 GHz) is actually hotter on average for these two benchmarks than the middle frequencies (1.8-2.4 GHz). This could be due to some sort of experimental error, as the benchmark runs

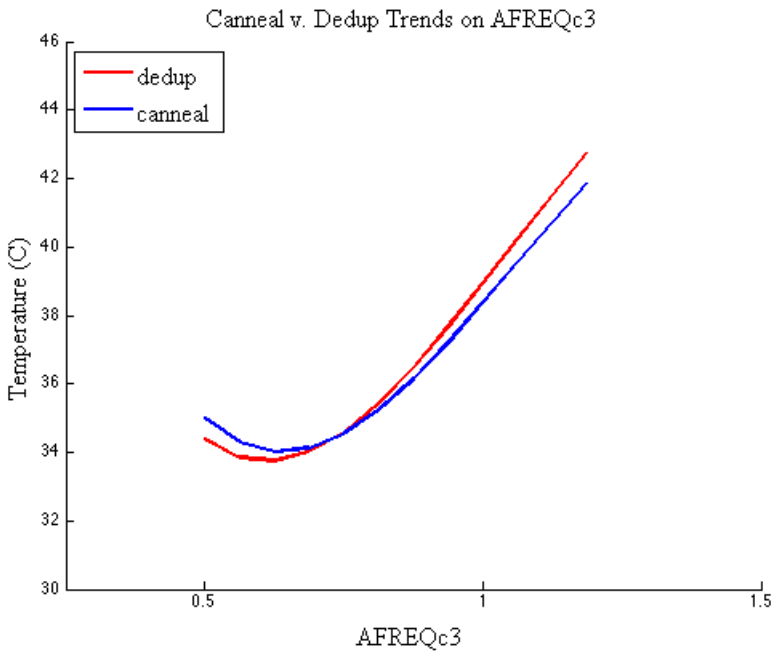

Figure 15: Neighboring core temperature vs. *AFREQc3* fitted trends for *canneal* and *dedup*

inherently could not be done in exactly the same conditions, even when they were done in the same room, in the same casing, with the same fan configuration, on the same day. The discrepancy could also be sensor error due to the measurement resolution.

Additionally, the two finalized models were used to predict values from the *x264* benchmark. The data gathered from *AFREQc3* from the *x264* benchmark was inserted into the polynomial for each benchmark and compared to the values of *wTEMPc2*, or the weighted running average temperature of Core 2. For the model created from the *canneal* data, the coefficient of determination (R^2) was 0.8894, and for the model created from the *dedup* data, the R^2 was 0.8959. Both of these values are close to the values for each of the benchmarks, and as a result, we can say that both of these models hold true for the *x264* benchmark as well.

CHAPTER 5

CONCLUSIONS

Beginning with the feature selection step, it is possible to draw conclusions about the relationship between temperature and the performance metrics. First, the factor analysis step indicated that the variance in the data is not heavily loaded on any one variable, which simply means that it is not possible to separate an important subset of performance metrics based upon their variances. It also leads to the conclusion that a number of the metrics are not independent. This is true, especially for the *AFREQc3* and *FREQc3* metrics and their derived variables. This is also true for the set of variables that are derived from instruction throughout (*EXECc3*, *IPCc3*, *INST*, *cINST*, etc.), and separately true for the subset of cache-based performance metrics (*L2HITc3*, *L2MISSc3*, *L2CLKc3*, *L3HITc3*, etc.).

The sequential feature selection step using stepwise selection allows for conclusions about this relationship to be drawn. The primary conclusion is that the strongest variable in predicting the neighboring core temperature across all benchmarks (compute and memory intensive) is the frequency at which the core is operating when in an active state. Additionally, the temperature tracks most strongly with a fitted third order polynomial of this frequency for both the *canneal* and *dedup* benchmarks. It is somewhat intuitive that the temperature-frequency relationship would follow something like a cubic function, given the measurements seen in this analysis. At lower frequencies, the temperature remains approximately constant, but in scaling to higher frequencies, the temperature increases rapidly.

This trend is surprising to find, as the analysis of the power consumed by the processing cores shows that both benchmarks' average power figures trend linearly with frequency. The cubic relationship found could be due to the physical phenomenon in the processor that is not captured by this analysis. From the simple resistance model in figure 1, it can be seen that if the temperature of the loaded core does not increase appreciably, the neighboring

core will not heat appreciably. At lower frequencies, the temperature stays approximately constant due to the temperature in the loaded core has not being perturbed enough to have an effect on the neighboring core This can be seen in figures 4 and 5. In those two plots, Core 3's temperature does not begin to increase appreciably until around the nominal frequency of the processor ($AFREQc3 = 1$, or 3.2GHz).

This does not explain why Core 3's temperature does not increase in a linear fashion, however. The simple model in figure 1 does not account for the heat capacity of the heat sink. In this case, it is likely that the heat sink can efficiently remove up to a certain amount of wattage without appreciably heating its entire volume above the ambient temperature. When the entire heat sink begins to heat above ambient, the temperature in Core 3 will rise. This is not modeled in the simple circuit because the heat sink is analyzed using the lumped capacitance model, which assumes that each "lump" has a constant temperature across its entire mass. For low power consumption this is likely not a valid assumption, as the heat sink was designed to cool a processor consuming 100+ W.

The energy and temperature results combine to yield another observation: the work done by the processor does not vary appreciably with a memory-intensive versus a compute-intensive workload. The average power results indicate that running *dedup* consumes slightly more power than *canneal* for a given frequency. If this extra power had been dissipated in the core, the increased power would have had an effect on neighboring core temperature. This can be seen by examining figure 6 and 15. As power increases with frequency, so does temperature. However, the increased power consumption in *dedup* does not appear to affect neighboring core temperature appreciably. As a result, it can be assumed that the extra power was consumed outside of the core over a sufficiently large area (such as across the large L3 cache seen in figure 2) or sufficiently far away from Core 2 to not affect the core's temperature.

Since this trend is present without accounting for what work the core is doing during a particular sampling period, it can be said that temperature of the neighboring core is heavily

dependent on frequency and not heavily dependent on the workload being executed on the loaded core. It can also be concluded that since the trends for *canneal* and *dedup* are so close to each other across the spectrum of frequencies that the two trends are essentially the same. This is an important observation that refutes the hypothesis of this thesis.

This conclusion also allows the profiling of the thermal coupling characteristics of this particular chip multiprocessor without needing to measuring the instruction throughput or cache behavior of a benchmark. If thermal coupling effects are to be minimized for a workload on this processor, there is a range of frequencies at which the loaded core should operate. This range is approximately 1.8 GHz through 2.4 GHz. This range is marked on figure 16 in green. This range represents the lowest temperatures for the neighboring core that were recorded during the testing of this processor, within the 1 °C resolution range of the digital temperature sensors. The data points called out in the figure are an average of the two models' predicted values for 1.8 GHz and 2.4 GHz.

Interestingly, this range does not coincide with the optimal energy-delay product range for either benchmark. For *canneal*, the optimal frequency was around 3.0 GHz for optimizing energy-delay product, while *dedup*'s results indicated that it should be run at the highest supported frequency to attain the minimized energy-delay product. This would indicate that thermal coupling effects do not play a large role in the energy-delay product when only one core is loaded.

5.1 Future Research

For future research this analysis can be run on other workloads to attempt to confirm the conclusions presented. Choosing two workloads that are representative of the spectrum of benchmarks is an approximation; it does not account for special cases where an often-used resource happens to be physically closer to a neighboring core. This could cause the neighboring core to heat more than what the models provided in this thesis would suggest. The reverse is also possible: if the resource is farther away, the neighbor may heat less.

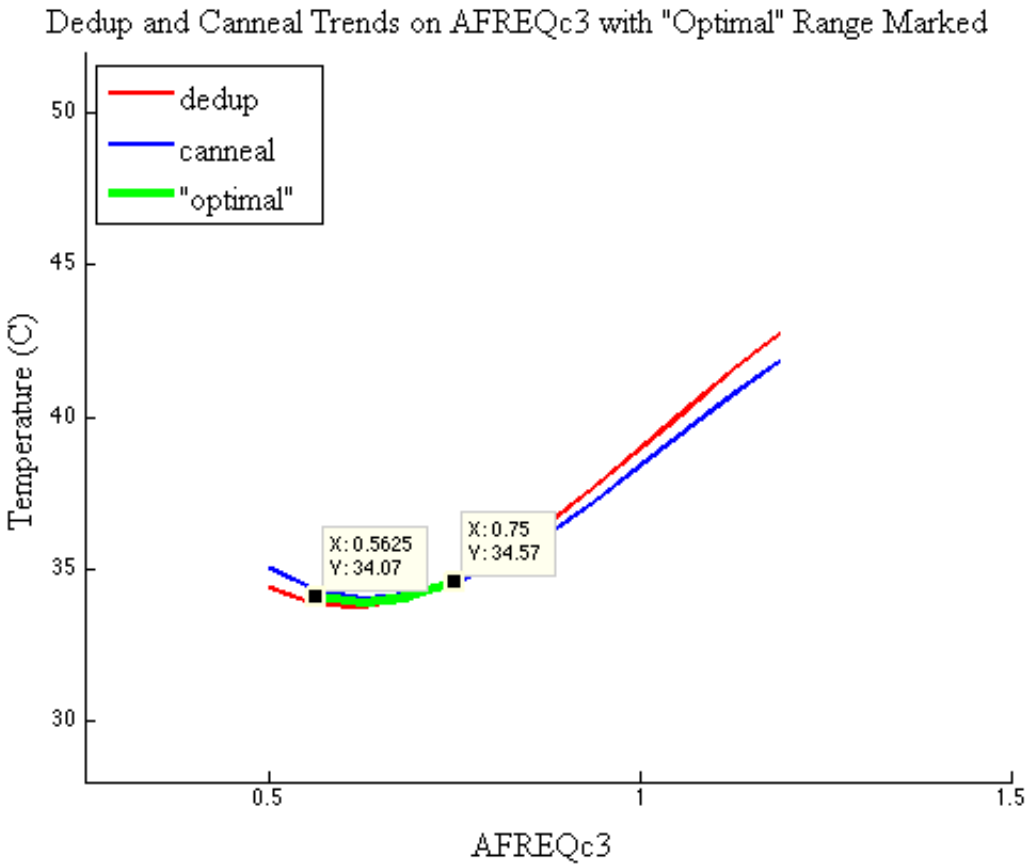

Figure 16: Neighboring core temperature vs. *AFREQc3* fitted trends, including the "ideal" range.

Another analysis that should be done is the analysis of multiple cores being simultaneously loaded, ideally with varied benchmarks. This analysis would take an appreciable amount of time to do, and would need to be done under rigorous conditions. Such a set of experiments would give more insight into how neighboring cores affect each others' temperatures. This would also allow for an analysis of how thermal coupling effects may effect the energy-delay product of a benchmark run.

To improve this model, it is possible that a cubic root function would fit this data well. However, there was no available software to perform a cubic root regression; it would have had to be done manually in steps in order to find the offset a in equation 11.

$$T = (x - a)^{1/3} + b \tag{11}$$

The b offset can be found with a simple regression, as it represents a constant term. To get the a term, sequential feature selection would need to be performed on the set of features that contain the equation below (without the b term) that contain test values of a. This is not an impossible process; it is simply time-intensive and the model found here was deemed to be good enough to draw conclusions from. However, it is still possible that a cubic root function could provide a better fit for this data.

In addition a logarithmic regression could be considered for analysis. This model may fit better than a simple cubic polynomial, and could provide a lower bound on the chip's operating temperature in this environment. A logarithmic regression was not done on this data due to time constraints, and due to the fact that the fit of the cubic polynomial was found to be sufficient for modeling the thermal coupling phenomenon found in this processor.

APPENDIX A

FACTOR ANALYSIS RESULTS

Table 4 contains the rotated principle components for the analysis with rotation of the *canneal* benchmark. The bold values indicate the largest values in that factor. This is to show that the first components do not appreciably load on a small subset of variables. These loadings indicate that there is not a simple way to isolate the variance contained in the measured data to a few features. Table 5 contains the respective eigenvalues of the first six principle components to show which components contain most of the variance. Table 6 shows the same principle components analysis from Table 4, but performed on the *dedup* benchmark. Similarly, Table 7 contains the eigenvalues from the analysis on the *dedup* benchmark. The results for loadings are largely similar to *canneal* in that they cannot be used effectively to inform the model building process, as the highest eigenvalue factors load on several variables.

Table 4: Rotated principle components for the analysis of *canneal*

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
FREQc3sqr	**0.99482**	-0.05046	0.23645	0.19387	0.08694	0.36923
AFREQc3sqr	**0.99482**	-0.05046	0.23645	0.19387	0.08694	0.36923
FREQc3cub	**0.98426**	-0.04839	0.22678	0.18689	0.0855	0.3545
AFREQc3cub	**0.98426**	-0.04839	0.22678	0.18689	0.0855	0.3545
FREQc3	**0.9945**	-0.05207	0.24494	0.19891	0.08795	0.38129
AFREQc3	**0.9945**	-0.05207	0.24494	0.19891	0.08795	0.38129
ACYC	**0.90318**	-0.04875	0.56263	0.32459	0.12805	0.35836
L3CLKc3	**-0.81145**	-0.31651	-0.37263	0.04379	-0.14635	0.1794
L2HITc3	0.01697	**0.95019**	0.45421	-0.34617	0.21821	-0.23671
L2CLKc3	-0.08759	**0.91258**	0.36744	-0.45954	0.14864	-0.22677
L3HITc3	0.02222	**0.95155**	0.42138	-0.40132	0.21058	-0.30047
IPCc3sqr	-0.18261	**0.93102**	0.31964	-0.15531	0.2882	-0.59352
IPCc3	-0.27901	**0.9436**	0.34043	-0.30433	0.24909	-0.5929
IPCc3cub	-0.09893	**0.80201**	0.24719	0.02426	0.30162	-0.56515
EXECc3	0.32492	**0.90399**	0.54462	-0.18493	0.34233	-0.31578
cINST	0.26197	**0.79197**	0.74396	0.07232	0.50368	-0.27821
INSTsqr	0.14085	0.45215	**0.92566**	0.17852	0.33283	-0.14356
INST	0.22162	0.62478	**0.94641**	0.10178	0.32227	-0.20432
WRITE	0.14614	-0.30669	0.1176	**0.95828**	0.08168	0.20298
READ	0.43326	-0.22889	0.32178	**0.79405**	0.05947	0.44522
cINSTcub	0.05134	0.19185	0.20741	0.06233	**0.96235**	-0.08152
cINSTsqr	0.1867	0.6632	0.63261	0.10613	**0.85895**	-0.25279
L2MISSc3	0.71717	-0.29244	0.08509	0.34177	0.00411	**0.82994**
L3MISSc3	0.68059	-0.41542	0.01459	0.3947	-0.02501	**0.82442**

Table 5: Eigenvalues of principle components for the analysis of *canneal*

Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
8.9658386	8.0796716	4.7587631	2.8590543	2.6934315	4.1142084

Table 6: Rotated principle components for the analysis of *dedup*

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
AFREQc3sqr	**0.93684**	0.0857	-0.00162	-0.10745	0.03445	0.02363
AFREQc3	**0.92538**	0.07392	0.00931	-0.0911	0.03882	0.01834
AFREQc3cub	**0.93638**	0.09491	-0.011	-0.12007	0.0305	0.02827
FREQc3cub	**0.98057**	0.25572	-0.11532	-0.37906	0.00681	0.1654
FREQc3sqr	**0.97954**	0.3198	-0.15882	-0.4872	-0.00088	0.2187
cINSTcub	**0.92007**	0.47745	-0.15275	-0.34809	0.09605	-0.04859
cINSTsqr	**0.94516**	0.52364	-0.20687	-0.47382	0.07655	0.04156
FREQc3	**0.92258**	0.42766	-0.208	-0.67063	-0.01393	0.29334
INSTsqr	**0.87042**	0.65979	-0.21381	-0.41078	0.11963	0.06818
ACYC	**0.88713**	0.41562	-0.15872	-0.62464	0.16768	0.26775
cINST	**0.90162**	0.5938	-0.28741	-0.6745	0.0405	0.18069
EXECc3	**0.88513**	0.69955	-0.33008	-0.64785	-0.04712	0.21343
INST	**0.87525**	0.70249	-0.31065	-0.63797	0.0629	0.20898
IPCc3cub	0.26216	**0.96977**	-0.32434	-0.41992	-0.03874	-0.06436
IPCc3sqr	0.30215	**0.99296**	-0.40922	-0.57575	-0.07349	0.04266
IPCc3	0.33238	**0.96078**	-0.46415	-0.72635	-0.1069	0.16569
L2MISSc3	-0.08963	-0.36662	**0.99072**	0.36512	0.45599	-0.44386
L3MISSc3	-0.10232	-0.3611	**0.9891**	0.36765	0.46128	-0.46057
L3CLKc3	-0.27484	-0.47802	**0.87293**	0.67567	0.25178	-0.57794
L2CLKc3	-0.36897	-0.62038	0.37881	**0.95374**	0.03824	-0.22137
L2HITc3	0.29675	0.49494	-0.70128	**-0.82942**	-0.27604	0.72065
READ	-0.05114	-0.20994	0.59379	0.24486	**0.9569**	-0.45388
WRITE	-0.09109	-0.30604	0.81883	0.29128	**0.83589**	-0.45627
L3HITc3	-0.02639	-0.10539	-0.52969	-0.17746	-0.45832	**0.938**

Table 7: Eigenvalues of principle components for the analysis of *dedup*

Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
11.630197	6.939557	5.655961	6.585575	2.473716	3.004173

APPENDIX B

FEATURE SELECTION RESULTS

Table 8 contains the summary of all the selection steps performed for the *canneal* benchmark using the forward stepwise selection method. This is to show mainly that the partial R^2 does not change the resulting model appreciably after the third feature is added to the model. Table9 contains the same information for the steps done while performing stepwise forward feature selection on the *dedup* benchmark. Similar results are obtained; however, for *dedup*, adding more features past the first two features does not appreciably change the model.

Table 8: Summary of selection steps for *canneal*

Step	Variable Entered	Variable Removed	Partial R-Square	Model R-Square	C(p)	F Value	Pr >F
1	FREQc3cub		0.8766	0.8766	112822	1278081	<0.0001
2	AFREQc3		0.0205	0.8971	64087.4	35939.2	<0.0001
3	FREQc3sqr		0.0171	0.9142	23616.1	35779.1	<0.0001
4	EXECc3		0.0054	0.9196	10853.4	12039	<0.0001
5	cINST		0.0014	0.9209	7648.63	3076.16	<0.0001
6	READ		0.0008	0.9217	5788.43	1804.24	<0.0001
7	L2HITc3		0.0006	0.9223	4427.34	1330.42	<0.0001
8	WRITE		0.0008	0.9231	2470.97	1931.95	<0.0001
9	IPCc3sqr		0.0004	0.9235	1581.49	883.76	¡.0001
10	IPCc3cub		0.0002	0.9236	1209.09	371.92	<0.0001
11	cINSTsqr		0.0002	0.9238	825.597	383.75	<0.0001
12	L2CLKc3		0.0001	0.9239	576.373	250.44	<0.0001
13	L3HITc3		0	0.9239	468.4	109.7	<0.0001
14	L3CLKc3		0	0.924	409.998	60.27	<0.0001
15	L2MISSc3		0	0.924	352.989	58.9	<0.0001
16	L3MISSc3		0.0001	0.9241	62.3731	292.54	<0.0001
17	INST		0	0.9241	37.5995	26.77	<0.0001
18	IPCc3		0	0.9241	19.4971	20.1	<0.0001

Table 9: Summary of selection steps for *dedup*

Step	Variable Entered	Variable Removed	Partial R-Square	Model R-Square	C(p)	F Value	Pr >F
1	AFREQc3cub		0.8698	0.8698	14288.6	184397	<0.0001
2	FREQc3		0.0084	0.8783	11575.2	1913.04	<0.0001
3	L3CLKc3		0.0075	0.8857	9175.51	1802.47	<0.0001
4	FREQc3cub		0.0055	0.8913	7402.47	1399.72	<0.0001
5	cINSTsqr		0.0023	0.8936	6657.77	601.63	<0.0001
6	L2HITc3		0.0004	0.894	6519.92	113.13	<0.0001
7	IPCc3		0.0007	0.8947	6302.45	178.7	<0.0001
8	AFREQc3		0.0005	0.8952	6136.98	137.03	<0.0001
9	AFREQc3sqr		0.0136	0.9088	1764.91	4112.43	¡.0001
10	FREQc3sqr		0.0015	0.9103	1300.24	445.83	<0.0001
11	IPCc3sqr		0.0012	0.9114	929.342	360.89	<0.0001
12	cINST		0.0013	0.9127	526.764	397.18	<0.0001
13	L3HITc3		0.0004	0.9131	399.365	127.62	<0.0001
14	IPCc3cub		0.0002	0.9133	348.508	52.23	¡.0001
15	L3MISSc3		0.0001	0.9134	304.803	45.23	<0.0001
16	WRITE		0.0004	0.9138	176.846	129.21	<0.0001
17		L2HITc3	0	0.9138	174.862	0.02	0.8999
18	L2CLKc3		0.0002	0.9139	128.026	48.64	<0.0001
19	L2MISSc3		0.0002	0.9142	53.6842	76.24	<0.0001
20	cINSTcub		0.0001	0.9142	39.2913	16.38	<0.0001
21	READ		0	0.9143	32.8903	8.4	0.0038
22	L2HITc3		0	0.9143	26.6488	8.24	0.0041

REFERENCES

[1] A. L. Shimpi and R. Smith, "The Intel Ivy Bridge (Core i7 3770k) review," April 2012.

[2] I. Paul, S. Manne, M. Arora, W. L. Bircher, and S. Yalamanchili, "Cooperative boosting: Needy versus greedy power management," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, (New York, NY, USA), pp. 285–296, ACM, 2013.

[3] Q. Xie, J. Kim, Y. Wang, D. Shin, N. Chang, and M. Pedram, "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pp. 242–247, Nov 2013.

[4] M. Janicki, J. Collet, A. Louri, and A. Napieralski, "Hot spots and core-to-core thermal coupling in future multi-core architectures," in *Semiconductor Thermal Measurement and Management Symposium, 2010. SEMI-THERM 2010. 26th Annual IEEE*, pp. 205–210, Feb 2010.

[5] M. Healy, H.-H. Lee, G. Loh, and S.-K. Lim, "Thermal optimization in multi-granularity multi-core floorplanning," in *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, pp. 43–48, 2009.

[6] H. Mizunuma, Y.-C. Lu, and C.-L. Yang, "Thermal coupling aware task migration using neighboring core search for many-core systems," in *VLSI Design, Automation, and Test (VLSI-DAT), 2013 International Symposium on*, pp. 1–4, 2013.

[7] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos, "Feedback thermal control of real-time systems on multicore processors," in *Proceedings of the Tenth ACM International Conference on Embedded Software*, EMSOFT '12, (New York, NY, USA), pp. 113–122, ACM, 2012.

[8] S. Corbetta and W. Fornaciari, "Exploiting thermal coupling information in mpsoc dynamic thermal management," in *Proceedings of the 26th International Conference on Architecture of Computing Systems*, ARCS'13, (Berlin, Heidelberg), pp. 50–61, Springer-Verlag, 2013.

[9] Intel Corporation, *Intel®64 and IA-32 ArchitecturesSoftware Developers Manual Volume 3B, Part 2*, June 2013.

[10] V. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring energy and power with papi," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, pp. 262–268, 2012.

[11] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, (New York, NY, USA), pp. 72–81, ACM, 2008.

[12] M. Bhadauria, V. M. Weaver, and S. A. McKee, "Understanding parsec performance on contemporary cmps," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC '09, (Washington, DC, USA), pp. 98–107, IEEE Computer Society, 2009.